SATIRE: A Software Architecture for Smart AtTIRE

Raghu K. Ganti
Department of Computer Science
University of Illinois, Urbana-Champaign
Urbana, IL 61801-2302
rganti2@cs.uiuc.edu

Praveen Jayachandran
Department of Computer Science
University of Illinois, Urbana-Champaign
Urbana, IL 61801-2302
pjayach2@uiuc.edu

Tarek F. Abdelzaher
Department of Computer Science
University of Illinois, Urbana-Champaign
Urbana, IL 61801-2302
zaher@cs.uiuc.edu

John A. Stankovic
Department of Computer Science
University of Virginia
Charlottesville, VA 22904-4740
stankovic@cs.virginia.edu

ABSTRACT

Personal instrumentation and monitoring services that collect and archive the physical activities of a user have recently been introduced for various medical, personal, safety, and entertainment purposes. A general software architecture is needed to support different categories of such monitoring services. This paper presents a software architecture, implementation, and preliminary evaluation of SATIRE, a wearable personal monitoring service transparently embedded in user garments. SATIRE records the owner's activity and location for subsequent automated uploading and archiving. The personal archive can later be searched for particular events to answer questions regarding past and present user activity, location, and behavior patterns. A short feasibility and usage study of a prototype based on MicaZ motes provides a proof of concept for the SATIRE architecture.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*data abstraction, information hiding*; D.4.7 [**Operating Systems**]: Organization and Design—*distributed systems, real-time and embedded systems*; D.4.8 [**Operating Systems**]: Performance—*measurements*; K.8.m [**Personal Computing**]: Miscellaneous

General Terms

Algorithms, Management, Measurement, Performance, Design, Experimentation

Keywords

Personal Monitoring, Smart Attire, Human Activity Identification

1. INTRODUCTION

Wearable personal monitoring devices (from pedometers and calorie counters to fall-detectors for elderly individuals) are being introduced increasingly to record and sometimes archive different user activities for medical, safety, personal, or entertainment reasons. The miniaturization of sensors and the emergence of wireless sensor networks in progressively decreasing form factors suggest that the trend of personal instrumentation will expand. Economic factors such as the approaching retirement of baby boomers and the increasing cost of health-care suggest viable applications of embedded personal instrumentation. An example application is one in which a patient needs to be monitored at home for a prolonged period of time by a care-giver. A software architecture is needed to facilitate development of such wearable personal monitoring applications, promoting their modularity and component reuse.

This paper presents the design, implementation, and evaluation of a wearable personal monitoring service, we call SATIRE. It allows users to maintain a private searchable record of their daily activities as measured by motion and location sensors, which are two of the most popular sensing modalities in personal instrumentation. This paper identifies the architectural requirements of such a service and develops a flexible and modular software solution on prototypical off-the-shelf hardware embedded non-obtrusively in a person's garment (in the padding of a winter jacket in our first prototype).

A major design goal of our service is transparency to the user. By instrumenting the user's attire non-obtrusively and implementing transparent information collection, storage, and upload, our service operates with no explicit input or maintenance required from the user. A related major goal is service longevity. To make the service practical, batteries should not need to be changed more than once a year, which means that they should last for at least one season [1]. We provide back-of-the-envelope calculations that show the feasibility of this lifetime based on proposed power management policies and expected usage patterns of (outer) garments.

Our current prototype records human activities and loca-

---

[1]Presumably a seasonal garment will not be worn more than one season per year.

tion using 2-axis accelerometers and GPS respectively, storing the measurements locally until they can be uploaded. This is achieved using a network of a few MicaZ motes woven into the lining and padding of a winter jacket. MicaZ motes are an off-the-shelf device of a size that is roughly half that of a modern cell-phone. By removing unnecessary components such as LEDs, the on-off switch, and AA-batteries (which can be replaced by smaller Lithium-based ones), it is feasible to embed the microcontroller-based device comfortably in many items of daily use such as footwear, belt-buckles, purses, and laptop cases, among other personal and wearable accessories. All recorded activity and location data can be subsequently uploaded through an *access mote* for private archival, from which past human activity and location information can be reconstructed. The access mote is a wireless gateway with Internet access that implements the SATIRE information upload protocol on top of an appropriate MAC-layer (802.15.4 in the case of MicaZ motes). We envision that a person using SATIRE would install an access mote at home (and at the office, if applicable). Cell-phones with appropriate radio and MAC-layer support can also implement access motes.

Observe that it is not our goal to provide continuous monitoring of the person at all times. It is best to achieve such continuity by collecting information from different sources such as instrumentation in the person's home, office, and garments. Our focus here is on information collected from the person's attire only. We identify several problems and corresponding architectural requirements in the development of attire-based personal monitoring systems. We categorize these problems into seven types, namely, (i) data collection and storage, (ii) data upload, (iii) data synchronization, (iv) power management, (v) reconstruction of activity logs, (vi) user interface, and (vii) security and privacy.

In this paper, we develop simple solutions to five of the seven problems, namely, data collection and storage, data upload, data synchronization, reconstruction of activity, and user interface. We also present a naive solution to protect the user's privacy. Finally, we provide a calculation of energy consumption under a simple energy management scheme to show that it meets longevity requirements.

Our work complements several smart in-home monitoring systems that have been developed and described in recent literature. Examples include the University of Virginia's Smarthouse [3], Georgia Tech's Aware Home [19], and University of Florida's Gator-Tech smart house [1]. These smart homes are built with a suite of non-invasive sensors placed in the environment of a person's residence, which monitor various human activities. Associated applications include provision of *virtual care*, where loved ones can be continuously informed about the activities and location of the user. Among other applications, Intel's *proactive health research* [2] group has built a wireless sensor network (WSN) for monitoring health care issues related to aging. Our wearable service can augment the data collected by a smart home by providing information regarding the user for periods of time when the user is not in the vicinity of instrumented infrastructure.

In addition to in-home systems, simple and cheap wearable accelerometric devices have been used in the past to identify various human activities and gestures. Context information such as walking, walking upstairs/downstairs and running is determined using accelerometers in [27], which is

used to display information as part of a Tourist guide application. Various techniques for identification of gestures and activities have been discussed in [30] and [33]. This work is complementary to ours. Algorithms for activity classification based on wearable sensor data can be incorporated into our architecture in a modular plug-and-play manner. The goal of this paper is precisely to provide a simple modular architecture into which different classification algorithms, communication protocols, and data filters can be integrated as components with well-defined interfaces.

The rest of the paper is divided into seven sections. Section 2 discusses reasons for our choice of hardware. Section 3 discusses a typical operational scenario and presents a flexible and modular software architecture for the system. Section 4 discusses the problems posed in the application development in greater detail and presents solutions to these issues. Section 5 describes two approaches that we employed for human activity identification. Detailed evaluation of the system is described in Section 6. Section 7 presents the related work and Section 8 concludes this paper giving directions for future work.

2. CHOICE OF HARDWARE

Motivated by the vision of having smart clothes which can log daily activities of a person, we first consider the hardware platform requirements of such an application. We divide the hardware components into (i) sensing, (ii) memory/storage, (iii) communication, (iv) processing, and (v) energy supply. This section summarizes the requirements in each of the aforementioned dimensions and consequently makes an argument for choosing a specific hardware platform.

Regarding sensing, we opted for acceleration and GPS sensors in the first prototype since we are primarily interested with activity and location monitoring. Acceleration data can be used to reconstruct activities based on motion patterns. We demonstrate such reconstruction using hidden markov models that are fed raw acceleration measurements to recognize activity types.

The memory and storage requirements arise from the expected sampling rate of user activity and the expected period of disconnected operation (i.e., operation where logging must be performed locally in the absence of an access mote). As described later in the paper, we show that 25 samples/s is sufficient to identify many activities using accelerometers. For a 2-axis accelerometer this gives 50 samples per second. We also show that 1 byte is enough to represent a sample. Under these conditions, each megabyte of storage can log roughly 6 hours of disconnected operation. We further show that compressing (e.g., run-length-encoding) intervals of stillness can significantly reduce storage requirements. Hence, storage capacity that is a significant fraction of a megabyte should be sufficient for logging most typical daily intervals of disconnected operation (e.g., the drive from home to office for someone with an access mote at both locations). Observe that to help save energy, logging should be done in flash memory such that logs are persistent across power-off intervals. This allows devices to execute at a low duty cycle when no activity is recorded. With the log (and code) residing in flash, the RAM requirements of our software become minimal (only a few kilobytes to hold data variables, a flash-page-size data buffer, a packet buffer, and a single execution context).

For communication, we opted for wireless support since,

in general, different (i.e., physically separate) pieces of attire should be able to collaboratively perform activity logging, not to mention the need for wireless information upload. A requirement on communication bandwidth stems from the interval it takes to upload the entire log (which is a significant fraction of a megabyte as derived above). It is desired that the upload interval be less than one minute such that transient encounters with an access mote are sufficient to upload all logged data. Allowing for retransmissions, this leads to an upload bandwidth requirement in the hundreds of kilobits per second.

The processing requirements of logging are minimal. The processing requirements and RAM requirements are consistent with the capabilities of a microcontroller. Finally, the device must be powered by a battery that lasts a season. A range of microcontrollers can be found that consumes power at a rate of 10-100 mW. At this rate, normal AA batteries can power the microcontroller for days. We show in this paper that aggressive power saving schemes can prolong that interval by approximately an order of magnitude getting close to the target battery lifetime.

Consistent with the above requirements, we considered several off-the-shelf microcontroller-based devices with wireless communication capabilities and an appropriately small form factor. A suite of such devices was developed in recent sensor networks research, such as *Mica2dot*, *Tmote Sky*, *Pluto* and *MicaZ*.

The features of the above sensor nodes are summarized in Table 1. These motes typically have a 8-bit microcontroller, and are powered by 3V batteries.

From Table 1, we observe that the MicaZ motes are a good choice. They have 2-axis accelerometers and a pluggable GPS module (available off-the-shelf). Usually, they use two AA batteries which with suitable power management techniques, can last for a full season. They use an 802.15.4 compliant radio which meets the upload bandwidth constraint. They also include a 512KB flash which is sufficient given the estimated logging requirements above. To make the design lighter and more ergonomic, AA batteries can be replaced with Lithium batteries. Lithium batteries with more than twice the capacity of AA batteries exist today, the use of which can extend the lifetime of the motes. Since this is a simple research prototype, we did not consider using Lithium batteries. The motes weigh about 18 grams (excluding the batteries) and the sensor boards are about the same weight as the motes. Whereas, two AA batteries weigh 46 grams, which is considerably heavier than the motes.

Observe that, the choice of our hardware is limited by the currently available off-the-shelf components. In the future, with better hardware, we will be able to create more applications.

3. APPLICATION SCENARIO AND SOFTWARE ARCHITECTURE

In this section, we describe a typical operational scenario and present a flexible and modular software architecture for SATIRE.

3.1 Application Scenario

A typical person wearing a SATIRE jacket goes about his/her normal daily activities as usual over the course of a season. During that time, the jacket records sensory data pertaining to the owner's whereabouts and activities. When the system comes in the vicinity of an access mote (a base mote connected to a PC), the logged data is uploaded reliably to a private repository associated with the person. Further, this data can be used to reconstruct the activities and locations of the person, which is explained in detail in the rest of the paper. This record can potentially act as a memory aid or help doctors in augmenting a patient's clinical information. Figure 1 gives a typical usage scenario of our system.
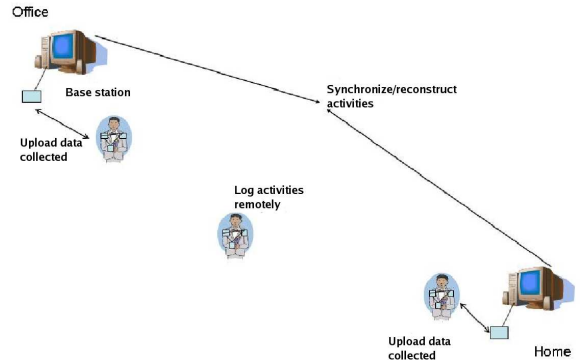

Figure 1: A typical operational scenario

3.2 System Architecture

The goals of our architecture are to allow flexible and modular development of personal instrumentation software. The architecture must accommodate a heterogeneous system that changes rapidly (e.g., due to introduction of new hardware, new system software, new sensor modalities, or new applications). This calls for an *application layer* to handle user interfaces for different applications. We are not only motivated by the need to upgrade or extend garment functionality over time (new software downloads to old garments are definitely within the realm of possibilities), but also by the fact that many different garment instances with different sensors should be able to leverage a common architecture. A future application might not focus on detecting human motion alone, but could possibly record vital signs of a person using sensors like EKG and pulse oximeter [21]. To enable this flexibility, we need a *parsing layer* to process the raw data generated by the introduction of new sensors. The new sensor modalities may give rise to changes in algorithms used for the reconstruction of the type of human activities performed. Moreover, given the same sensory data, different algorithms may be developed to identify different activities. Hence, the need for an *interpretation layer* to handle different algorithms that interpret the processed sensor data.

Based on the above characteristics of the architectural framework, we propose a layered architecture as shown in Figure 2, which advocates modularity and transparency, where user involvement is kept to a minimum. This architecture is two-dimensional, the first dimension is the *PC/laptop* and the second dimension is the *motes*. The architecture is layered to abstract out common concerns behind a standard interface. Note that, the application, operating system and the hardware layer abstractions on the motes side have been well developed by the sensor network community. TinyOS [4], a widely used operating system for sensor networks,

Characteristics	Mica2dot	Tmote Sky	Pluto	MicaZ
Off the shelf sensing modalities	2-axis accelerometer, light, microphone	humidity, light, temperature	accelerometer	accelerometer, GPS, light, temperature
Storage	512 KB	1 MB	1 MB	512 KB
Radio bandwidth	19.2 Kbps	250 Kbps	250 Kbps	250 Kbps

Table 1: Summary of characteristics of the Mica2dot, Tmote Sky, Pluto, and MicaZ motes

provides the necessary abstraction to enable introduction of new sensor modalities. We propose data collection and storage, upload, and data synchronization protocols at the operating system layer. Further, we introduce *layer 3*, which is the sensor specific protocols.
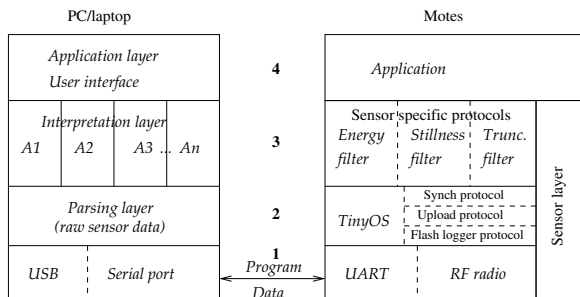

Figure 2: Architectural framework

We now describe each layer in detail following the journey of a given sensor data item. The item originates at the sensor layer which passes it directly to the application at *layer 4*. The data is then passed down to *layer 3*, which is the filter layer. It implements data processing algorithms which are common to multiple applications. Such algorithms might include stillness detection (used to trigger a power-saving mode and reduce the amount of flash-consumption), averaging, and other simple signal conditioning techniques. The results are passed down to *layer 2*, which includes the operating system and those protocols that are fundamental to the working of the system, such as the upload protocol and the data logging protocol. This layer provides an interface that higher-level applications and filters can wire to using nesC (a programming language used for TinyOS) wiring conventions. At the bottom of the stack is *layer 1*, which implements communication device drivers. It will not be described further in this paper. The layers on the mote side are executed in real-time when the data is collected.

On the PC side, data is uploaded at *layer 1* when the motes on a garment come in contact with an access mote. Such data are passed to *layer 2*, which is the parsing layer. It collects and parses the raw bytes uploaded by the motes and stores them as files of *(time, sample)* pairs or other standard data formats. When a new sensor is used, we simply plug-in its corresponding parsing module into this layer. For example, GPS data is stored in the NMEA format. The name of a file determines the type of sensor responsible for the data within. Observe that changes in data representation will only change the parsing layer and not affect any other layer. The data collected by *layer 2* could be used in many kinds of applications for different purposes (other than activity identification). The interpretation layer (*layer 3*) is the off-line data processing layer. It interprets the data collected using various data mining or statistical and signal processing techniques. Different algorithms for reconstruction of the type of human activity can be plugged into this layer, thus enabling

easy upgrade of garment functionality. *Layer 3* abstracts the data-processing algorithms (named *A1*, *A2*, ..., *An* in the figure) and presents their results in a standard format as predicates asserted to *layer 4*, the application layer. Subsequently, a user can generate a query such as "Where did I eat lunch last time I visited New York City?" A query engine or a logical inference engine at *layer 4* can process those queries. Multiple *layer 3* algorithms can be executed on the stored data, thus enriching the pool of predicates that can be asserted regarding the user, facilitating more advanced inference at *layer 4*.

Based on the above generalized architectural framework, we developed the SATIRE system. Figure 3 details the specific modules which we have developed and the interactions between each of these modules. Observe that the user interface layer is further divided into three interacting sub-layers. These are the *web-interface, web-server* and *MySQL database server*.
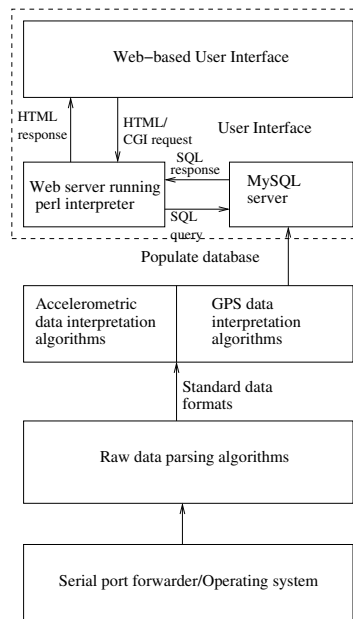

Figure 3: Detailed architecture of the PC side, also giving the interaction details of the layers.

The user feeds her query using the web-interface. The web-server interacts with the web-interface and generates dynamic HTML content to answer the user's queries and performs necessary interpretation of the queries. The MySQL database server stores the users' profiles. The current database table for activity storage consists of the fields *person, activity type, start time* and *end time*. The *person* field indicates the user with whom the data is associated, the *activity type* is the nature of the activity performed (such as sitting, walking, climbing stairs, and eating) and the start and end times of these activities are given by the *start time* and *end time*

fields respectively. Adhering to the general architecture we have presented, we have shown that a simple modular system can be developed efficiently.

4. PROBLEM DISCUSSION AND PROPOSED SOLUTIONS

We describe the problems faced during the development of each layer of our system. In the operating system layer on the motes, the main problems include data collection and storage, data upload to the PC, data synchronization, and power management. The other layers on the motes have been well developed by the sensor network community. On the PC side, the main problems are reconstruction of activity logs in *layer 3*, developing a user friendly interface and addressing privacy and security issues in *layer 4*. We will describe these problems in further detail and present simple yet efficient solutions to solve them.

4.1 Data Collection and Storage

4.1.1 Problem Discussion

In a typical operational scenario, the system will collect data periodically and store it in the flash memory. A MicaZ mote has 512 KB of flash memory, which is used for data recording purposes. Hence, we observe that a single sensor sampling at the rate of 30 Hz, generating 2 bytes per sample will consume the flash memory in approximately four hours. Increasing the number of sensors used will consume the flash even faster! A simple proposition to reduce the amount of flash consumed is to reduce the sampling rate, but this would be inadequate as the data values recorded cannot be used to reconstruct the activities. We conducted simple experiments to identify an ideal sampling rate. We use walking as the base activity to come up with an efficient sampling rate. Every step taken is a peak in the accelerometric data. By manually counting the number of peaks and then matching it with the number of steps taken, a given sampling rate can be justified. Our experiments were conducted on two different subjects. From these experiments, we observed that the number of footsteps taken were between three and six every second. Hence, from Nyquist-Shannon's sampling theorem, the minimum (lossless) sampling frequency is at least $12\ Hz$. Since this is the bare minimum required, we propose to use the sampling rate of $25\ Hz$ for each axis of the accelerometer. Thus, we need other methods to reduce the amount of flash consumed in order to increase the disconnected time of operation of the system.

4.1.2 Proposed Solution

We propose two different methods to reduce the amount of flash used without loss of the precision of data collected. Both are different data compression algorithms based on the observations we made during the deployment of the system. The first method, termed the *truncate filter* takes advantage of the output range of the accelerometer values. We observed that the 10 bit output from the MicaZ accelerometer does not vary widely and the output value range, for normal human activities, can be usually accommodated within 8 bits! This doubles the disconnected time of operation of the system. An alternative solution to this problem is to record only the differences between successive sensor readings, which will again fit within 8 bits given the range of

data values. The second method takes advantage of the fact that all the values need not be recorded if the accelerometer output does not change. This means that we do not need to record any data values when the clothing is still, as there is no activity taking place. We observed that a typical user's jacket was still for a majority of the time. Our observations were based on deployment of the system in two separate scenarios. We propose the *stillness filter*, which identifies whether the accelerometric signals indicate stillness. Observe that, stillness is indicated only if both the x and y axes data values do not change over a brief period of time. This filter calculates the energy of a discrete *difference* signal periodically for stillness detection. The total energy of a discrete signal over n values is defined as:

$$E\ =\ \Sigma_{i=1}^n x_i^2\ \ where\ x_i\ is\ a\ sample\ value \qquad (1)$$

Similar to run-length encoding, at the end of a stillness interval, a special separator value is inserted in the log, indicating that the jacket has been still and the number of samples for which it has been still is recorded.

4.2 Data Upload

4.2.1 Problem Discussion

An important part of the system is to upload the data collected to a server through an access mote. Three separate issues are to be addressed as part of the upload protocol, which are as follows:

- The rate at which the upload occurs, which directly affects the amount of flash available.

- The upload transparency, which relieves the user from the hassle of pressing a button or consciously making a gesture to upload data.

- Reliability of upload, Loss of data packets can lead to incorrect interpretation of activities or lack of data for certain periods of time. As expected, when the user moves away from the access mote, the packet reception probability goes down.

We develop a new protocol optimized for our application scenario, which meets the above goals.

4.2.2 Proposed Solution

Our protocol combines ideas from various data dissemination protocols like *Deluge* [15] and *PSFQ* [34]. It achieves the goal of reliable, transparent and fast upload. Transparency is achieved by using a *beaconing* scheme. The motes in the system send beacons periodically, which are ACKed by the base if the mote is in the range of the base. Reliability is achieved using a NACK scheme. We tweaked the payload size in TinyOS and the number of packets sent every second, to come up with an optimal data rate to send data as fast as possible. We use the CSMA MAC protocol which comes with the TinyOS networking stack. Further discussion on the parameter selection is presented in Section 6. Our protocol makes sure that only one mote is communicating with the base at a given time to minimize collisions and increase throughput. This is ensured as follows. The base listens to the beacons from the motes and replies with a *send data* packet to the first mote from which it hears a beacon. After sending this packet, the base enters a state where it does not reply to any further beacons. The base resets its state

after a timeout period, during which it does not receive any data packet from the mote which it had granted permission to send.
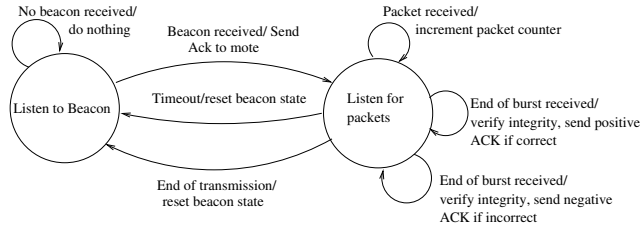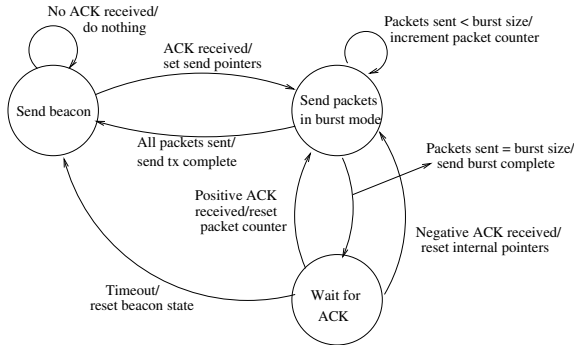

Figure 4: State diagram for the base

Figure 5: State diagram for the motes on the person

Figures 4 and 5 summarize the upload protocol which we present in the form of state diagrams. The annotations on each arrow have the form X/Y, where X indicates the event which has occurred and Y describes the action to be performed on occurrence of this event.

4.3   Data Synchronization

4.3.1   Problem Discussion

We need a mechanism by which it is possible to correlate the activities recorded on different parts of the body. This problem can be termed as the *data synchronization* problem, where each data item collected needs to be temporally correlated with data items collected on other motes. This problem has been addressed in [36]. The scheme presented in [36] needs a base station in the vicinity of the data collection nodes, which SATIRE cannot use, and thus a new data synchronization scheme is needed. Apart from the above scheme, there have been several time synchronization protocols which synchronize the clocks on motes [23], [13]. However, recording absolute time values leads to a considerable overhead in the flash. The periodic message exchanges contribute an additional overhead.

4.3.2   Proposed Solution

To maintain temporal correlation among the data values collected on different motes, the simple beaconing scheme used for data upload is also used to synchronize data streams on the different motes in the network. Each beacon is identified by a beacon number and each mote is associated with a set of unique beacon numbers, which wrap around sufficiently far away in time not to cause ambiguity. Received

beacon numbers of other motes are recorded in real-time in the flash mid-stream along with a separator, to differentiate them from data sample values. When the streams are reconstructed on the PC, identically numbered beacons are aligned to the same time reference. The only overhead in our method is that of recording the values of beacons in the flash. Beacons samples occur several orders of magnitude less frequently than data, which makes their overhead acceptable.

4.4   Power Management

4.4.1   Problem Discussion

Motes will need battery replacement after a rather short amount of time. The typical lifetime of a mote which is *on* for the entire period of time is about seven days. With continuous logging and radio communication, the lifetime may be further reduced. Replacement of batteries every week is cumbersome and cost ineffective. Hence, a power management scheme is necessary to extend the lifetime of the system. An acceptable design goal for an outer garment in our opinion is to last for about three months.

4.4.2   Proposed Solution

We propose to use a simple duty cycle based scheme. In this scheme, a mote goes to *sleep* after it detects a brief period of stillness. It wakes up after n seconds and checks whether or not stillness continues. If the mote determines that it is in motion, it starts logging data. Otherwise, it goes to sleep again. During this cycle, the mote keeps track of the amount of time it has been sleeping and logs this information when the stillness interval terminates.

Figure 6 shows the statistics for the amount of time three motes were still and the time they were recording motion over a complete monitoring experiment. Figure 6 suggests that most of the time the jacket is not in motion, which implies that the person is moving around for only a brief amount of time with the jacket on.

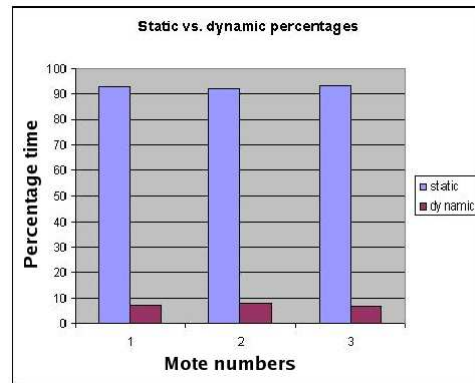**Figure 6: Percentage of time motes were still (static) and motes were recording (dynamic)**

Figure 6 validates that our proposed power management technique will significantly extend the battery lifetime of the system. The figure indicates that low-power operation is possible at least 90% of the time.

If we assume a 5% duty cycle during low-power operation, the lifetime of the system can be extended seven times, as

can be seen from Equation 2. In Equation 2, P_d is the power consumption when the mote operates as described above, P_n is the power consumption when the mote is active all the time, and d is the duty cycle (in our case, it is 0.05). The power consumption of the mote in a low-power state is assumed to be negligible. This gives us $\frac{P_d}{P_n}$ to be about seven, which translates into an increase in the lifetime of the jacket from one to about seven weeks. This is close to the season-long goal we set out for our smart jacket.

$$P_d \quad = \quad 0.1 \times P_n + 0.9 \times (d \times P_n) \qquad (2)$$

As for the GPS mote, it draws a higher current than a normal sensor and will last for about half a day if it is left on continuously. The GPS mote takes about seven seconds to obtain a fix (usually). In the active state, the GPS mote obtains a fix every minute and sleeps for the rest of the minute. When the jacket is still, it does not obtain a fix. In this scenario, the GPS mote lasts for seven weeks ($\frac{60}{0.1 \times 7} \times 0.5$ days), which is close to our season-long goal.

Observe that, the introduction of the above power management scheme is problematic to the beacon based data synchronization. This is because, when a mote sends a beacon, the other motes could be asleep and fail to record the synchronization point. We resolve this problem by adopting the following simple method. Initially, all the motes are awake. When the first mote sends its beacon, the other motes listen, and synchronize their clocks to this beacon. Due to clock drifts of upto $40\mu s$ per second, synchronization will be lost after a certain time duration. If we assume that, the mote is awake for t ms after sending its beacon (before going back to sleep), then synchronization will be lost if the drift with another mote is more than t ms. This will happen after $\frac{t \times 1000}{40}$ seconds. To achieve re-synchronization, when a mote fails to receive a beacon from another mote, it will remain awake until it receives a beacon from that particular mote. If t is significant (in the order of a hundreds of milliseconds, which is a fair assumption, as the motes will remain awake for a few hundred milliseconds in order to check for stillness), then re-synchronization needs to be done once in a few thousands of seconds.

4.5 Reconstruction of Activity Logs

4.5.1 Problem Discussion

Until now, we have described the issues arising on the motes' side. The system is not useful if it cannot reconstruct the activities done in the past or if it is not possible to associate a *real* time with these activities and locations. The problem of reconstruction of activities performed and location inference is two-fold. First, we must determine the *type of activity performed*. Examples include `walking`, `sitting`, `climbing stairs`, `typing`, and `reading`. Second, we need to associate a *time* with these activities. The first problem has been addressed in the literature in the past and several algorithms have been proposed [30], [27], [33] and [25] to interpret the human activities and gestures from raw accelerometric data. We compare the typical approach of using feature vectors to a new Hidden Markov Model (HMM) based approach.

4.5.2 Proposed Solutions

Several features of a signal have been introduced in [30] and [33] for the purpose of activity identification. A subset of these features can be mapped onto a multidimensional feature space which can be used for activity identification. For example, the energy of the difference signal of the x and y accelerometer axes is useful to identify some activities. Figure 7 plots a two-dimensional feature space, where each dimension is the energy of the difference signal of the corresponding accelerometer axis when passed through an *exponential weighted moving average (EWMA)* filter. The figure shows a map of the EWMA values of x versus that of y for different activities. Outliers have been eliminated by means of averaging.
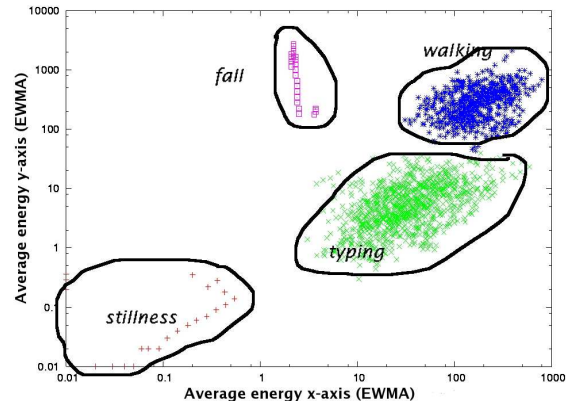

Figure 7: Map of X vs Y EWMA values for certain activities (Data obtained by performing specific experiments)

From Figure 7, we can see that different activities have clearly identifiable regions. If new activities are identified that fall in the same region in the mentioned two-dimensional feature space, we can use additional features to identify differences between such activities. In Section 5, we compare this identification approach to one that uses HMMs.

Given the raw data from the motes, we need to reconstruct the *real world time* when a sample value was collected. To do so, we go backwards in time. When data is uploaded by the motes to an access mote, we assume that the last value uploaded is the most recent value and assign it a time which is the current time on the PC. Given the number of sample values between two timestamps, we can get an estimate as to when a particular sample value was collected. This method only gives an approximate estimate due to clock drift on the motes.

4.6 User Interface

4.6.1 Problem Discussion

Presenting the data to the user in a user-friendly manner and yet providing sufficient and exact details is an interesting problem. Observe that, multiple visualizations of the same data are necessary because different users have different tastes. Access to one's profile should also be round-the-clock and not limited to the user's home or office computer.

4.6.2 Proposed Solution

A natural solution to address a round-the-clock availability of the user data archive is to use the Internet. As discussed in Section 3, we use a web-server and a database management system (DBMS) for answering the queries of

the user. The DBMS and the web-server are maintained at a trusted remote centralized location, to which the users can connect using the Internet. A simple user-interface with authentication is provided, so that the user can log-in, upload activity logs, and query her profile's database. Each user owns a private location on the DBMS server, where her data is encrypted and stored. Alternatively, users with privacy concerns can host their data on their home computer (assuming it is constantly online). This solution addresses the issue of privacy partially, as it secures access to the user's data.

4.7 Privacy and Security

4.7.1 Problem Discussion

The issue of *privacy* and *security* comes to the forefront as the sensory data collected and archived may contain sensitive information which belongs to the individual who has been using the application. Hence, the system design should address security concerns through the development of secure channels of communication and not allowing unauthorized access to information.

4.7.2 Proposed Solution

A basic privacy scheme which uses an access matrix for checking who can access what will suffice for our initial prototype. This will be part of the *layer 4* on the PC side. Security issues can be addressed by using an authentication based scheme for initiating a data upload and using a secure communication channel for the actual data upload to take place. Such a scheme is presented in [18]. We have not implemented secure communication in the current prototype.

5. HUMAN ACTIVITY IDENTIFICATION

In this section, we compare approaches taken for human activity identification. Our first approach was to extract a set of features from the accelerometric signal and use these to identify activities which were well spread out in the feature space. This is similar to the approach taken in [30], and we claim no novelty in this regard. Feature vectors have also been used in speech recognition [31]. This approach is described in more detail in Section 5.1.

We found that the accuracy of this approach was poor when we used it to identify multiple activities. We provide a further discussion of this in Section 6. To overcome the drawback of using a static feature vector model, we used a dynamic Hidden Markov Model (HMM) to solve the problem. HMMs have been used in several machine learning and speech recognition applications [26]. This approach is explained in Section 5.2.

5.1 Feature Vector based Identification

Our feature space consists of features such as average, standard deviation, root mean square, range, integral, temporal variation, and rotational direction. Temporal variation is the sum of absolute euclidean distances between accelerometric vectors of any two successive time instances. Rotational direction gives an idea of clockwise/anticlockwise rotational movement during the activity [30].

An activity can be represented as a static n-dimensional vector, where n is the number of features, obtained through representative training data. To identify a given activity, features are extracted from the raw accelerometric data, and

a least error match with the representative feature vectors is found.

5.2 HMM based Identification

A Hidden Markov Model (HMM) is a statistical model where the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters, based on this assumption. Although HMMs have been used in the past for machine learning and speech recognition applications, to the best of our knowledge there has not been much work on using HMMs for identification of human activities using accelerometers [24]. In [24], HMMs have been used to identify complex wood workshop activities. In contrast to this, our emphasis is on identifying a broader range of mundane activities.

A HMM is characterized by the following:

- N: The number of hidden states

- M: The number of distinct observation symbols per state

- $A_{N \times N}$: State transition probability distribution

- $B_{N \times M}$: Observation symbol probability distribution for each state

- $\Pi_{N \times 1}$: Initial state distribution

There are two phases to using a HMM. The first is a training phase, where the HMM learns the model parameters that maximize the probability of observing the representative data set. The second is the inference phase, where the probability of an observation sequence to be classified is calculated given the HMM model.

To solve the problem of human activity identification, we use an ergodic (every state of the model can be reached from every other state) and discrete observation HMM. For details of the training and inference techniques, the reader is referred to [26]. For each activity, we have a $N(N = 10)$ state HMM, where a range of raw accelerometric data are mapped onto a single observation symbol of the HMM. We use a total of 120 observation symbols ($M = 120$) per state. We use the well known Baum-Welch technique, which is based on an Expectation Maximization (EM) algorithm, for training the HMM for each activity. To identify an activity, we use a Forward-Backward procedure, which given a HMM model, estimates the probability that the observation sequence is generated by this model. Using this procedure, the probability that the observation sequence is generated by the HMM for each of the activities is calculated. The activity that yields the highest probability is chosen as the activity represented by the observation sequence.

6. EVALUATION

We split the evaluation into three parts. The first part of our evaluation deals with micro-benchmarks which affect the overall performance of our system. The next subsection is concerned with the evaluation of the performance of the protocols developed in Section 4. We then present a high level evaluation of the overall performance of the system.

6.1 Micro-benchmarks

In order for the upload protocol to function well, we evaluate the parameters affecting communication performance and use the best empirically determined values as the parameter settings for our system.

First, observe that data upload can use one of two methods. In the first one, packet upload is clocked. Every time the timer fires, a packet is sent. An alternative method to clocked packet transmission is to send each packet immediately after the previous packet completes transmission, which is signified by the *sendDone* event in TinyOS. This is called a split phase command, since the *send* command returns followed by the asynchronous signal *sendDone*, when the operation is successful.

Figure 8 plots the sent and the received data rates when the number of bytes in the packet are varied for a split phase send. We notice that the maximum achievable data rate is about 35 Kbps at the receiver. The advertised radio data rate of MicaZ motes is 250 Kbps and that of the serial port is 57.6 Kbps. We conducted an experiment to measure the maximum data rate achievable on the serial port, and we were able to achieve about 50 Kbps, which makes the serial port the bottleneck. The base mote has only 512 KB of flash memory, which makes it imperative for it to forward a packet on the serial port as soon as it receives the packet from the embedded sensor motes. Hence, the base mote has to interleave the radio reception and forwarding of a packet on the serial port, further reducing the achievable data rate. This accounts for the 35 Kbps data rate we were able to achieve.
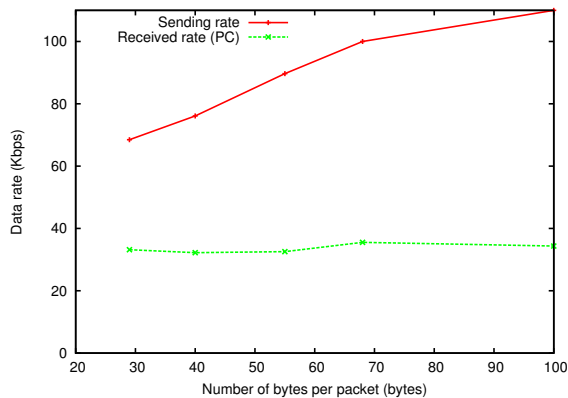

Figure 8: Comparison between sent and received data rates using split phase send

We also notice from Figure 8 that due to the bursty sending rate of split phase send, high losses are incurred. Hence, we chose to use a clocked send approach. Here, we control the data rate of the sender using a timer, so that the sending data rate is maintained at about 30 Kbps. Figure 9 plots the sent and received data rates for varying number of bytes in a packet. We notice that for packet sizes greater than 68 bytes, we were able to achieve nearly lossless transmission.

We observe that while sending data, read operations occur from the flash which prohibits any writes to the flash during the send interval. Hence, it is important that the sensory data collected be kept in volatile memory until they can be written. The amount of volatile memory available for data buffering is very small (less than 4KB). This imposes a condition where reads and writes must be interleaved to reuse
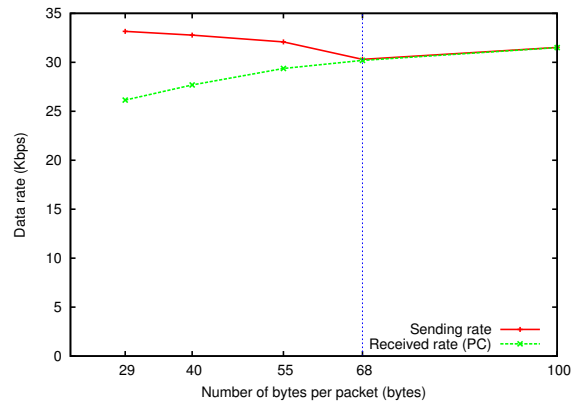

Figure 9: Comparison between sent and received data rates using clocked send

the buffer space. Our upload protocol presented in Section 4 takes care of this problem by checking after every send (i.e., read) burst for any data collected from sensors and writing such data into the flash in a write burst. Hence, upload time (i.e., read burst time) becomes a dominating factor that determines the buffering requirements in our system. Note that, there are two different bottlenecks in our upload. One is at the access mote attached to the PC, which is constrained by the rate at which data can be sent to the PC. The other is on the mote in the jacket, where data needs to be buffered during the upload, and reads/writes to the flash need to be interleaved.

Figure 9 shows that nearly lossless transmission can be achieved for data rates of about 30 Kbps. The percentage of packets received for different data rates is plotted in Figure 10 for MicaZ motes. The data rate is varied by changing the timer period. From this figure, we observe that the percentage of packets received falls sharply for data rates greater than 30 Kbps.
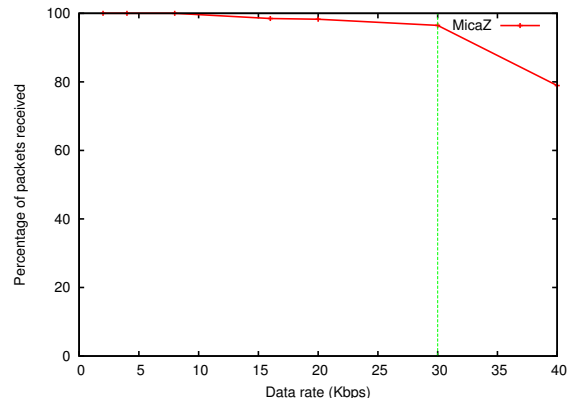
Figure 10: Percentage of packets received as the data rate is increased (by decreasing the timer period)

We conclude from Figures 8, 9, and 10 that a good set of values to send data without incurring heavy losses for MicaZ motes is about 68 bytes per packet with about 55 packets sent every second, giving a net data rate of roughly 30 Kbps, which we deem sufficient for our purposes.

It is a known fact that as the distance from the base is increased, the packet loss also increases [35]. For the simple ARQ/NACK based scheme we presented in Section 4, Figure

Figure 11: Percentage overhead cost of reliability as the distance from base is increased

Figure 12: Amount of flash used vs. Filters applied

Figure 13: Deviation of empirical sampling rate from the theoretical sampling rate

11 correlates the cost of reliability in terms of percentage overhead with increasing distance from the base.

We observe that the overhead is quite high at large distances. From our experiments, we also observed that the time taken for the reliable upload protocol to send packets successfully is quite high when the distance from the base is increased beyond 25-30 ft. This distance, however, may be sufficient in practice. For example, it is consistent with the dimensions of a walk-in closet or a cubicle, where a jacket and an access mote can communicate. Also note that the entire upload time is very small. Even substantial overhead will extend this time by only a few seconds. Hence, the cost of reliability is not an issue.

6.2 Protocol Performance Evaluation

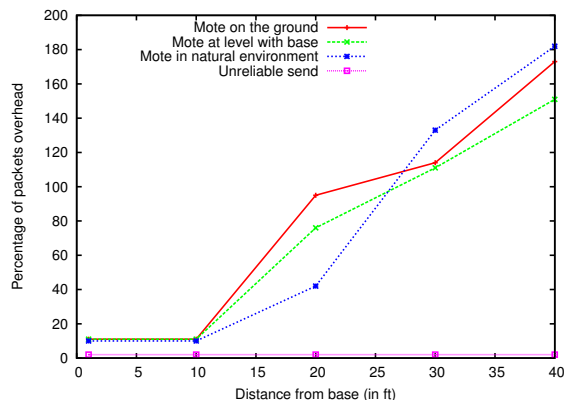This section evaluates the protocols we developed for the system. Our prototype includes six motes, five of which log the x and y axes accelerometer data and the sixth mote runs a GPS module. All the motes are placed unobtrusively inside the lining of a heavy winter jacket, which has been specially created for these motes. Two motes were placed on each arm (one below and one above the elbow) and one mote was placed close to the waist, with all the y-axes pointing downwards when in standing position (with the arms pointing down). The GPS mote can be placed anywhere, but the GPS receiver is placed facing outwards for better satellite reception.

Our architecture enables the easy introduction and removal of filters. Figure 12 shows the amount of flash used[2] when corresponding filters are applied. Data is presented from snapshots of two different experiments involving two different people.

We observe from Figure 12 that the amount of flash saved by employing filters is quite high, indicating that a person's activities in a day are mainly confined to bursty intervals. Filtering can also reduce the amount of energy consumed. Since the number of times we write to the flash has decreased, it extends the disconnected operation period and the battery life.

We presented a method to use beacons to reconstruct the time a particular sample value was collected. Figure 13 plots the deviation of the theoretical sampling rate (which is 25Hz for each accelerometer axis in our case) from the actual sampling rate maintained by the system as calculated from bea-

con timestamps over the duration of an entire experiment.

We observed from the raw data summarized in Figure 13 that the deviation from the actual sampling rate is higher when the system is in a disconnected mode of operation for a longer time (not shown in figure). This shows that the clock drift of the motes affects the correctness of reconstruction of time. If the system is near the base most of the time, the clock drift does not affect the calculated sampling rate which means that the data synchronization protocol will be more accurate. Hence, our method is approximate if the time of disconnected operation is large and is quite accurate when this time is smaller.

6.3 Overall System Performance

In this subsection, we present a performance evaluation of the various sub-systems in SATIRE.

6.3.1 Human Activity Identification

We classify the activities into *stationary* and *non-stationary*. An activity is classified as *stationary*, when the energy of the difference signal (as described in Section 4.1) summed over a period of time is lower than a threshold. Otherwise, the activity is classified as *non-stationary*.

Different activity identification algorithms can be easily plugged into the interpretation layer of our architecture. We used two different interpretation techniques, namely identification using feature vectors and using HMM. In all our activities, the data set was obtained by conducting each activity for a period of ten minutes. A sample of one minute was used to train the HMM for each activity. Ground truth

---

[2]the value is greater than 512KB since we use the flash as a circular buffer

was verified by manually recording the activity at a given time instant. Data sets were obtained for two different users.

Figures 14 and 15 plot the accuracy of detecting a set of seven *stationary* activities using both the feature vector and the HMM approaches for two different users. The activities considered were `sitting`, `reading`, `typing`, `lying down`, `standing in an elevator`, `writing`, and `eating with a fork and knife` (only for user 1). The feature vector approach performed poorly when compared to the HMM approach. This is due to the fact that the feature vector approach does not consider the sequence in which the motion is performed, but rather relies on a set of static features. Figure 16 plots the accuracy of identifying three *non-stationary* activities, namely, `walking`, `walking with an umbrella`, and `climbing stairs` for a user using HMM and feature vector approaches.
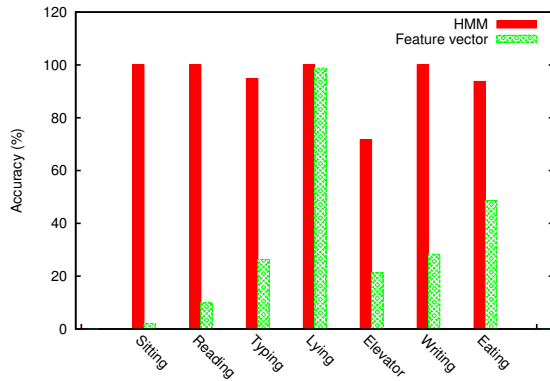

Figure 14: Accuracy of stationary activity identification using HMMs and feature vectors for user 1
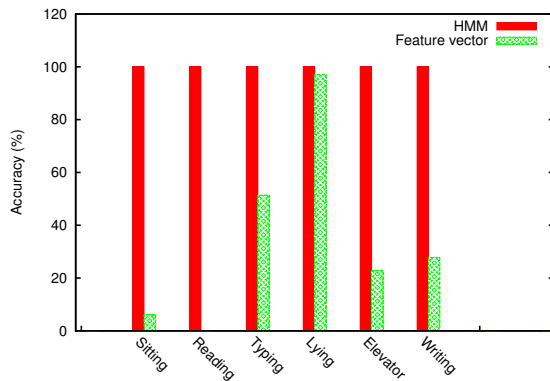
Figure 15: Accuracy of stationary activity identification using HMMs and feature vectors for user 2

We then trained the HMM using one user's data and tested the accuracy of identifying the other user's stationary activities. This is plotted in Figure 17. We observed that `sitting` was classified with very high accuracy, while `reading` and `typing` were not identifiable (zero accuracy for `reading`). This suggests that, `reading` and `typing` are specialized activities for each user, while `sitting` is a general activity.

A new user can use the jacket to identify a set of pre-trained general activities. To identify new activities or to

Figure 16: Accuracy of non-stationary activity identification using HMMs and feature vectors

Figure 17: Accuracy of stationary activity identification without user specific training

improve the accuracy of identification of existing activities, the user can specially train the jacket to suit his/her needs.

6.3.2 Location Tracking using GPS

We added a GPS sensor mote to our system by introducing a module in the parsing layer of our architecture. The GPS mote can be used to track the location of the user when he or she is not occluded from the GPS satellites. We conducted experiments to track the location of a user. The experiment was conducted over a period of seventy-five minutes, when the user went home from his department and returned after about thirty-five minutes. The GPS locations obtained from this experiment were superimposed on a map of the area and are shown in Figure 18 to visually present the route taken by the user. From the figure, we notice that the period of time the user was at home is displayed as a dense cluster of points, towards the south-west portion of the route.

For the same experiment, we plot the speed-time graph in Figure 19 and the activity recognized over time in Figure 20. The user first walked for about 15 seconds and then took an elevator. After which, we identified that the user was walking with a speed of about 1.5 m/s for about 25 minutes. This, we noticed was the time he walked to his home. Then, he was stationary for about 35 minutes, after which he walked back to the department at an average speed of 1.5 m/s. The user again used the elevator and walked for about 15 seconds (to his office).

We conducted another experiment, for which the location details and the speed-time plots are shown in Figures 21 and

Figure 18: Expt. 1: Location tracking using GPS

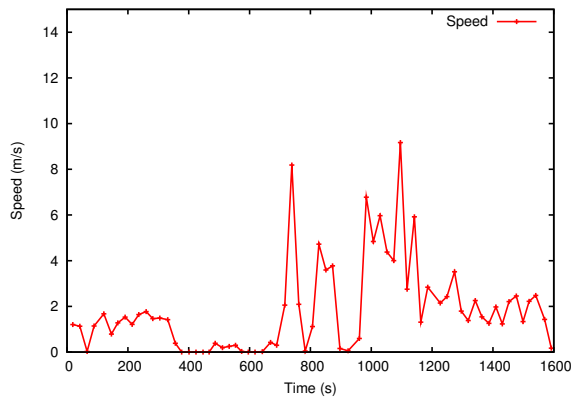**Figure 19: Expt. 1: Speed vs. Time of user**

Figure 20: Expt. 1: Activities of user over time

Figure 21: Expt. 2: Location tracking using GPS

22, respectively. From these figures, we were able to deduce that the user was walking in `Region 1` (with an approximate speed of 1.5 m/s) and was stationary for about 5 minutes in `Region 2`. At about a time of 60 seconds, we observe that the speed of the user was 0 m/s. This was because the user was waiting to cross a busy road. The user's speed was found to fluctuate between 0 and 10 m/s during times 700 and 1150 seconds, indicating that the user was in a vehicle that made frequent stops. In fact, this was found to be true as the user was traveling in a campus bus, which made frequent stops. From time 1150 seconds onwards, the user was found to be walking at about 1.5 m/s.

In conclusion, the vision of smart attire (at least for outer garments such as winter jackets) is quite realizable assuming that hardware is developed that can be comfortably embedded in the lining, which is protected (by proper enclosures) from moisture, shocks, and chemicals (as in dry-cleaning).

7. RELATED WORK

We divide the related work into two sub-sections namely *gesture recognition interfaces* and *wearable computers*.

7.1 Gesture Recognition Interfaces

Gesture recognition and motion tracking have received much attention in previous literature. Several efforts have focused on hand motion recognition. For example, hand motion capture was described using vision techniques [9]

and infrared [29]. Generally, one of the most widely investigated approaches for human body motion tracking is to use cameras. Human motion is reconstructed from tracking features such as lights attached to human joints [16]. All the above methods require the use of external or head-mounted cameras to track motion, which are cumbersome to use. In contrast, our work focuses on human motion recognition through the use of accelerometric data.

Accelerometers have been frequently used for performing motion tracking. For example, an accelerometer-based gesture recognition interface is presented in [8], which acts as a pointing device and menu selector, among other options. In [6], a low-cost and low-power wearable motion tracking system is developed, based on integrated accelerometers. Signature verification using accelerometers is presented in [7] that measures amplitude, phase and frequency of pen acceleration signals. Hand motion recognition using accelerometers received further attention. An *acceleration sensing glove* is presented in [25] that can detect and translate finger and hand motions into computer interpreted signals. A hand gesture recognition method is described in [32] using a hand shape Data Glove for acquiring a series of hand shapes in a continuous motion for gesture recognition. These techniques can be imported into our general architecture.

Gravity and magnetic field have often been used for position estimation. Magnetic field, Angular rate and Gravity sensors (MARG sensors) are used in [5] to determine the posture of an articulated body. In [28], a micro detec-

121

Figure 22: Expt. 2: Speed vs. Time of user

tor with integrated micro electro-mechanical sensors, such as accelerometers and magnetometers, was used for motion tracking and online simulation.

7.2 Wearable Computers

Several specialized wearable computing systems have been described in recent literature. One common goal of many systems has been to attain context awareness. Self-guided tour applications received some attention. For example, a *cyberjacket* was described in [27] for a tourist guide system. In [12], a system that combines overlaid 3D graphics of augmented reality with mobile computing is developed to present information about a university campus. Other context-awareness devices include a wearable jacket developed in [11] which uses knitting techniques and stretch sensors to measure upper limb and body movement and contains a 2-axis accelerometer. MIThril [10] describes a PDA-centric architecture for wearable computers. A method to recognize the context-awareness of the user from stream inputs from sensors worn by the person is described in [14]. This system uses sensors mounted on several circuit boards and attached to a utility belt worn by the user. A system with accelerometers on pants attached to a laptop to interpret the raw sensor data using Kohonen maps and machine learning techniques is presented in [20]. In [17], a low power, distributed platform that combines acceleration and magnetic field sensors in a wearable, hierarchical network is presented. The system is divided into subnetworks assigned to body parts that are connected to a local bus with a dedicated master. This system can be used for context sensing and gait analysis. Smart Attire introduces the next generation of wearable computing systems in which wired networks and centralized processing are replaced with wireless sensors individually equipped with their own microprocessors, memory, and radio devices. This decentralization offers more flexibility, scalability, and independence within the computing platform.

Exciting research is also being done in the area of e-textiles [22], where several prototypes of fabrics are being developed, that make embedding sensors and processing elements in clothing easy.

8. CONCLUSION AND FUTURE WORK

In this paper, we presented a novel way of recording sensory data generated by outdoor human activities using a wireless sensor network embedded unobtrusively in the clothing of a person. These data are collected remotely and uploaded to a base station opportunistically for reconstruction of activities. In doing so, we identified the major issues and problems involved in building such a system. We implemented efficient solutions to these problems, which include remote data logging (with increased disconnected operation as the major issue), upload protocols for the raw sensory data collected, and reconstruction of the type/time of activities performed. Each of these protocol implementations was evaluated in a real world deployment by conducting several experiments.

We also presented a flexible and modular software architecture for future development of smart attire systems that simplifies introduction of new sensors and new algorithms. We are yet to implement the power management protocol, privacy, and security protocols presented in this paper on the real system. Security and privacy issues have to be addressed due to the sensitive nature of the data collected. Issues such as distributed processing have not yet been addressed in this paper. Our future work will develop an operating system for use over WSNs embedded in clothing and for development of various algorithmic techniques to identify different kinds of human activities. Another branch which we are yet to explore is the usefulness of raw data for medical applications.

9. ACKNOWLEDGMENTS

10. REFERENCES

[1] Gator-tech smart house, http://www.rerc.ufl.edu/.
[2] Intel's proactive health research lab, http://www.intel.com/research/prohealth/.
[3] Medical automation research center (marc), http://www.marc.med.virginia.edu/.
[4] Tinyos, http://www.tinyos.net/.
[5] E. R. Bachmann, R. B. McGhee, X. Yun, and M. J. Zyda. Inertial and magnetic posture tracking for inserting humans into networked virtual environments. In *Proc. of the ACM symposium on virtual reality software and technology*, pages 9–16. November 2001.
[6] R. Barbieri, E. Farella, L. Benini, B. Ricco, and A. Acquaviva. A low-power motion capture system with integrate accelerometers (gesture recognition applications). In *Proc. of the IEEE consumer communications and networking conference*, pages 418–423. IEEE, January 2004.
[7] R. Baron and R. Plamondon. Acceleration measurement with an instrumented pen for signature verification and handwriting analysis. *IEEE Transactions on instrumentation and measurements*, pages 1132–1138, December 1989.
[8] A. D. Cheok, K. G. Kumar, and S. Prince. Micro-accelerometer based hardware interfaces for wearable computer mixed reality applications. In *Proc. of IEEE international symposium on wearable computers*, pages 223–230. IEEE, October 2002.

[9] J. Cui and Z. Sun. Visual hand motion capture for guiding a dextrous hand. In *Proc. of the IEEE International conference on automatic face and gesture recognition*, pages 729–734. IEEE, May 2004.

[10] R. DeVaul, M. Sung, J. Gips, and A. S. Pentland. Mithril 2003: Applications and architecture. In *Proc. of the IEEE international symposium on wearable computers*, pages 4–11. IEEE, October 2003.

[11] J. Farringdon, A. J. Moore, N. Tilbury, J. Church, and P. D. Biemond. Wearable sensor badge and sensor jacket for context awareness wearable sensor badge and sensor jacket for context awareness. In *Proc. of the IEEE international symposium on wearable computers*, pages 107–113. IEEE, October 1999.

[12] S. Feiner, B. MacIntyre, T. Hollerer, and A. Webster. A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. In *Proc. of the IEEE International Symposium on Wearable Computers*, pages 74–81. IEEE, October 1997.

[13] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *Proc. of ACM SENSYS*, November 2003.

[14] A. R. Golding and N. Lesh. Indoor navigation using a diverse set of cheap, wearable sensors. In *Proc. of the IEEE International Symposium on Wearable Computers*, pages 29–36. IEEE, October 1999.

[15] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proc. of ACM SENSYS*, November 2004.

[16] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14:201–211, 1973.

[17] H. Junker, P. Lukowicz, and G. Troster. Padnet: wearable physical activity detection network. In *Proc. of the IEEE International Symposium on Wearable Computers*, pages 244–245. IEEE, October 2003.

[18] C. Karlof, N. Sastry, and D. Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *Proc. of ACM SENSYS*, November 2004.

[19] C. D. Kidd, R. J. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. Mynatt, T. E. Starner, and W. Newstetter. The aware home: A living laboratory for ubiquitous computing research. In *Proc. of the International Workshop on Cooperative Buildings*, October 1999.

[20] K. V. Laerhoven and O. Cakmakci. What shall we teach our pants? In *Proc. of the IEEE International Symposium on Wearable Computers*, pages 77–83. IEEE, October 2000.

[21] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *Proc. of the International Workshop on Wearable and Implanatable Body Sensor Networks*, April 2004.

[22] D. Marculescu. E-textiles: Toward computational clothing. *IEEE Pervasive Computing*, 2(1):89–95, January-March 2003.

[23] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *Proc. of ACM SENSYS*, November 2004.

[24] D. Minnen, T. Starner, J. A. Ward, P. Lukowicz, and G. Troster. Recognizing and discovering human actions from on-body sensor data. In *Proc. of the IEEE International Conference on Multimedia and Expo*, July 2005.

[25] J. K. Perng, B. Fisher, S. Hollar, and K. S. J. Pister. Acceleration sensing glove (asg). In *Proc. of the IEEE international symposium on wearable computers*, pages 178–180. IEEE, October 1999.

[26] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, February 1989.

[27] C. Randell and H. L. Muller. The well mannered wearable computer. In *Proc. of Personal and Ubiquitous computing*, volume 6, pages 31–36, February 2002.

[28] Z. Rong, Z. Zhaoying, Y. Ting, and S. Xufeng. Micro measurement system for tracking human body motion. *Qinghua Daxue Xuebao/Journal of Tsinghua university*, 43:1464–1467, 2003.

[29] Y. Sato, K. Oka, H. Koike, and Y. Nakanishi. Video based tracking of user's motion for augmented desk interface. In *Proc. of the IEEE international conference on automatic face and gesture recognition*, pages 805–809. IEEE, May 2004.

[30] H. Sawada and S. Hashimoto. Gesture recognition using an acceleration sensor and its application to musical performance control. *Electronics and Communications in Japan*, 80(5):452–459, 1997.

[31] S.-A. Selouani, H. Tolba, and D. O'Shaughnessy. Auditory-based acoustic distinctive features and spectral cues for robust automatic speech recognition in low-snr car environments. In *Proc. of the American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 91–93, 2003.

[32] T. Takahashi and G. Kishino. A hand gesture recognition method and its application. *Systems and Computers in Japan*, 23:33–48, 1992.

[33] P. H. Veltink, H. B. J. Bussmann, W. de Vries, W. L. J. Martens, and R. C. V. Lummel. Detection of static and dynamic activities using uniaxial accelerometers. *IEEE Transactions on Rehabilitation Engineering*, 4:375–385, December 1996.

[34] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. Psfq: A reliable transport protocol for wireless sensor networks. In *Proc. of the ACM International Workshop on Wireless Sensor Networks and Applications*, September 2002.

[35] A. Woo, T. Tony, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. of ACM SENSYS*, November 2003.

[36] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proc. of ACM SENSYS*, November 2004.